LabVIEW Academy
12. óra – event, property node
Event-Driven Programming

• Events – Definition
• Event-Driven Programming – Definition
• Polling Versus Event Structures
• Parts of an Event Structure
• Configuring the Event Structure
• Caveats and Recommendations
Events

- Events originate from the user interface, external I/O, or other parts of the program.
- Events do things TO event sources.
  - Example: Value change happens TO front panel controls.

**Event** — An asynchronous notification that something has occurred
Event-Driven Programming — Method of programming where the program waits for an event to occur before executing one or more functions.
Polling versus Event Structures

• Polling
  – Method of event-based programming where a loop must continually run code to check if changes have occurred.
  – Polling the front panel requires a significant amount of CPU time.
  – Polling can fail to detect changes if they occur too quickly.

• Event Structures
  – Events in Event structures eliminate the need to poll the front panel.
  – Benefits of using Event structures:
    – Reduces the CPU requirements of the program.
    – Simplifies the block diagram code.
    – Guarantees that the block diagram can respond to all interactions the user makes.
Using Event Structures for Event-Driven Programming

• An Event structure works like a Case structure with a built-in Wait on Notification function.

• Use an Event structure to handle user-interface (static) events such as:
  – Pressing a button on the mouse.
  – Pressing a key on the keyboard.
  – Changing the value of a numeric control.
Parts of an Event Structure

• Event Selector Label—Identifies the event case viewed.
• Timeout—Specifies time in ms to wait for events. Default value is -1 (indefinite).
Parts of an Event Structure (continued)

- Event Data Node—Identifies the data LabVIEW provides when the event occurs; similar to the Unbundle By Name function.
- Event Filter Node—Identifies the subset of data available in the Event Data node that the event case can modify.
Using an Event Structure

In general, place Event structures inside While Loops.

– Event structures handle exactly one event per iteration of the While Loop.

– Event structures sleep when no events occur.
Configuring the Event Structure

- Use a dialog box to configure each event by right-clicking the Event structure border and selecting **Edit Events Handled by This Case** from the shortcut menu.
Edit Events Dialog Box

- Configured Events
- Event Sources
- Events

Click OK to accept your changes. Click the Add or Remove button to define another event specifier to be handled by this case or remove the currently selected event.
Notify and Filter Events

Notify Events (green arrow)
User action has already occurred and LabVIEW has processed the event.

Filter Events (red arrow)
User action has already occurred and LabVIEW has NOT processed the event.
Filter events allow you to override default behavior for event.
Configure and Use Events

• Demonstrate configuring and using an Event structure.
Exercise 1-2

Concept: Event Structure

- Convert a polling-based application to an Event structure-based application.
Exercise 1-2
Concept: Event Structure

• How many times did the loop run with polling?
• How many times did the loop run when you changed the VI to use an Event structure?
Caveats and Recommendations

• Avoid using an Event structure outside of a loop.
• Place only one Event structure in a loop.
• Avoid configuring two Event structures for the same event.
• Use a Value Change event to detect value changes.
• Keep event handling code short and quick.
• Place Boolean control terminals inside an event case for latched operations to work properly.
Summary—Quiz

1. Which of the following buffer data?
   a) Queues
   b) Events
   c) Local Variables
Summary—Quiz Answer

1. Which of the following buffer data?
   a) Queues
   b) Events
   c) Local Variables
Summary—Match the Following

1. Obtain Queue
   a. Destroys the queue reference

2. Get Queue Status
   b. Assigns the data type of the queue

3. Release Queue
   c. Adds an element to the back of a queue

4. Enqueue Element
   d. Determines the number of elements currently in the queue
Summary—Match the Following Answer

1. Obtain Queue
   - a. Destroys the queue reference

2. Get Queue Status
   - b. Assigns the data type of the queue

3. Release Queue
   - c. Adds an element to the back of a queue

4. Enqueue Element
   - d. Determines the number of elements currently in the queue
3. Which of the following are valid data types for queues?
   a) String
   b) Numeric
   c) Enum
   d) Array of Booleans
   e) Cluster of a String and a Numeric
3. Which of the following are valid data types for queues?
   a) String
   b) Numeric
   c) Enum
   d) Array of Booleans
   e) Cluster of a String and a Numeric
Summary—Quiz

• 4. The Event structure handles only one event each time it executes.
  a) True
  b) False
Summary—Quiz Answer

• 4. The Event structure handles only one event each time it executes.
   
   a) True
   b) False
VI Server Architecture

- VI Server Architecture
- Properties and Methods
- VI Class Hierarchy
VI Server Architecture

• The VI Server provides programmatic access to LabVIEW.

• Use the VI Server to:
  • Programmatically control front panel objects & VIs
  • Dynamically load and call VIs
  • Run VIs on a computer or remotely across a network
  • Programmatically access to the LabVIEW environment and editor (Scripting)
Properties and Methods

Properties — Single-valued attributes of the object: read/write, read only, write only

Properties include color, position, size, visibility, label text, and label font.

Methods — Functions that operate on the object

Methods include reinitializing values to default and exporting graph images.
VI Server—Class Hierarchy

Class

Control
  Property: Visible
  Method: Reinitialize to Default

Sub-Class

Boolean
  Property: Boolean Text

Array

Numeric

Object

Stop Button
  Property Values
  Label Text: Stop Button
  Visible: Yes
  Boolean Text: Stop
VI Server—Class Hierarchy

VI

Application

Generic

Etc.

GObject

Control

Boolean

GraphChart

Numeric

Cluster

Etc.

Generic

GObject

Control

Boolean
Property Nodes

- Definition
- Creating Property Nodes
- Execution Order
Property Nodes

- Property Nodes read and write the properties of an object. Property Nodes can:
  - Change the color of a chart plot.
  - Disable and enable controls.
  - Get the location of a control or indicator.
- Property Nodes allow you to make these modifications programmatically.
- Use Context Help to get information about properties.
- There are two types of Property Nodes.
  - Implicitly linked
  - Explicitly linked
Creating Property Nodes

- Create a Property Node for a front panel object.
Execution Order

- Property Nodes can have multiple properties.
  - Properties execute from top to bottom.
Exercise 3-1
Display Temperature and Limits

• Use Property Nodes to change the properties of front panel objects programmatically.
Exercise 3-1
Display Temperature and Limits

• What would be the Disabled state of the controls if the user aborts the VI instead of stopping the VI? How can we prevent this from happening?
Invoke Nodes

• Definition
• Control Methods
• VI Methods
Invoke Nodes

Invoke Nodes call methods or actions on objects.

» Get VI Version
» Print VI panel
» Reinitialize All to Default
  – Invoke Nodes perform actions on referenced items such as VIs and controls.
  – Most methods have parameters.
  – Use Context Help to get information on methods.
  – There are two types of control Invoke Nodes.
    • Implicitly linked
    • Explicitly linked
Control Methods

To create an implicitly linked Invoke Node:

1. Right-click the control terminal on the block diagram and select Create » Invoke Node.
2. Select a method from the submenu.
   – Examples of control methods:
     • Reinitialize to Default
     • Export Image
VI Methods

• Use a VI Server Reference to associate an Invoke Node with the current VI.

• To create a VI method:
  1. Place an Invoke Node on the block diagram.
  2. Right-click and select **Select Class** to choose a class.
  3. Right-click again and select **Select Method** to choose a method.
Exercise 3-2
Customizing the VI Window and Export Data

• Learn how to affect the attributes of a VI by using Property Nodes and Invoke Nodes.
Exercise 3-2
Customizing the VI Window and Export Data

What other front panel attributes and actions might you want to control programmatically from the block diagram?
• Implicitly and Explicitly Linked Property Nodes
• Create SubVIs
  – Use the Create SubVI Tool
  – Create Manually
• Select a VI Server Class
Control References

Implicitly Linked Property Node

Explicitly Linked Property Node
Control References

• A control reference is a reference to a front panel object.
• Wire control references to generic Property Nodes.
• Pass control references to subVIs.
Creating a SubVI

To create explicitly linked Property Nodes in a subVI:

1. Create your VI.
2. Select the portion of the block diagram that will be in the subVI.
3. Select Edit»Create SubVI. LabVIEW automatically creates the control references needed for the subVI.
4. Customize and save the subVI.
Creating a SubVI

SubVI Front Panel

SubVI Block Diagram
Create Control References Manually

SubVI Front Panel

SubVI Block Diagram

Ctl Refnum

Ctl Refnum

Enable Status

Enable Status

Digital

Numeric

0
Selecting the VI Server Class

• After you place a Control Refnum on the front panel of a subVI, specify the VI Server class of the control.
  – Right-click and select **VI Server Class** from the shortcut menu.
  – Alternatively, drag a control into a control refnum to specify the type.

• The class specifies the type of control references that the subVI accepts.
Exercise 3-3
Create SubVIs for Common Operations

• Use control references to create a subVI that modifies graph or chart properties.
Notice that several refnums are created but not closed in this exercise.

- Typically, you close refnums to avoid performance and memory issues.
- VI Server Reference constants associated with controls or This VI are special cases. You do not need to close these implicit references.
Summary—Quiz

1. For each of the following items, determine whether they operate on a VI class or a Control class.
   a. Format and Precision
   b. Visible
   c. Reinitialize to Default Value
   d. Show Tool Bar
Summary—Quiz Answer

1. For each of the following items, determine whether they operate on a VI class or a Control class.
   a. Format and Precision: **Control**
   b. Visible: **Control**
   c. Reinitialize to Default Value: **Control**
   d. Show Tool Bar: **VI**
2. You have a Numeric control refnum in a subVI. Which control references could you wire to the control refnum terminal of the subVI?

   a. Control reference of a Knob
   b. Control reference of a Numeric Array
   c. Control reference of a Thermometer indicator
   d. Control reference of an LED
Köszönöm a figyelmet!